◆招待論文◆

Performance Evaluation of ROHust Header Compression (ROHC) over Unidirectional Links using DVB-S Testbed

DVB-S を使用した片方向通信路における ROBust Header Compression (ROHC) の性能評価

Way-Chuang Ang
Research Officer, School of Computer Sciences, Universiti Sains Malaysia
アン ウェイ チャン
マレーシア科学大学計算機科学部研究員

Tat-Chee Wan
Senior Lecturer, School of Computer Sciences, Universiti Sains Malaysia
ワン タ チー
マレーシア科学大学計算機科学部専任講師

Kotaro Kataoka
Doctoral Program, Graduate School of Media and Governance, Keio University
片岡 広太郎
慶應義塾大学大学院政策・メディア研究科後期博士課程

Chee-Hong Teh
Research Officer, School of Computer Sciences, Universiti Sains Malaysia
テ チー ホン
マレーシア科学大学計算機科学部研究員

Unidirectional Lightweight Encapsulation (ULE) was introduced to carry IP data over Unidirectional Link (UDL) using Digital Video Broadcasting via Satellite (DVB-S) system to overcome efficiency problems in satellite data communication. This paper introduces a method to further improve the efficiency of IP packets transmission over satellite communication system using RObust Header Compression (ROHC). ROHC is a framework to compress headers of IP packets. This paper presents a study of performance characteristics of ROHC over DVB-S via an actual satellite link. The analysis of results showed that header compression yields significant improvement in terms of data throughput when the payload sizes of IP packets are less than 512 bytes, typical of VOIP and other realtime traffic.

Unidirectional Lightweight Encapsulation (ULE) は、衛星通信回線の片方向通信路（Unidirectional Link, UDL）上で Digital Video Broadcasting via Satellite (DVB-S) を用いて効率的な IP データ通信を行うための仕組みである。本稿では、RObust Header Compression (ROHC) を用いて、より効率的に衛星通信回線上で IP データ通信を行う手法を提案した。ROHC は IP ヘッダの圧縮手法である。実験により衛星通信回線上で DVB-S と ROHC を利用した際の通信効率を測定し、解析した。その結果、VOIP 等のリアルタイム通信に多く使われる、512 バイト未満の IP パケットに適用することで、通信効率が大幅に改善することが判明した。

Keywords: DVB-S, ROHC, ULE, MPEG2-TS, UDL Mesh

1 Introduction

A technique to setup a mesh network over Unidirectional Links (UDL) was proposed in our work [1] to achieve better spectral efficiency, whereby each peer node participating in the network broadcasts its traffic to its peers. This topology allows for lower latency as the communication path between peers is only one hop. The star topology networks as seen in Figure 1 incur an additional hop because traffic between leaf nodes has to go through a central hub. In addition, star topology networks are not as spectrum efficient since their resultant spectrum usage is greater than that of UDL mesh based topology [1]. The UDL mesh based topology is logically represented in Figure 2. Each site listens to the transmissions from its counterparts and broadcasts its data to every partner sites, whereas in a star topology, each site only communicates with the central hub. The single point of failure at the central hub in the star topology will bring down the whole network.

Digital Video Broadcasting via Satellite (DVB-S), a standard within the family of DVB standards, is commonly used to deliver audio/video content over satellite links, but it can be used to deliver IP packets as well. DVB-S system uses Moving Picture Experts Group 2 Transport Stream (MPEG2-TS) frame to carry data. MPEG2-TS is a stream of fixed size frames consisting of 4 bytes of header and 184 bytes of payload. Traditionally, IP packets were encapsulated using Multi-Protocol Encapsulation (MPE) before they are encapsulated within payload section of MPEG2-TS frames. In addition to a more efficient UDL mesh network topology, Unidirectional Lightweight Encapsulation (ULE) [2], a lightweight encapsulation format proposed by IETF, was used as the replacement for MPE. A study by Teh, et. al. [3] showed that ULE was more efficient than MPE. Efficient utilization of bandwidth is very important when its cost is expensive as is the case for satellite communications technology.

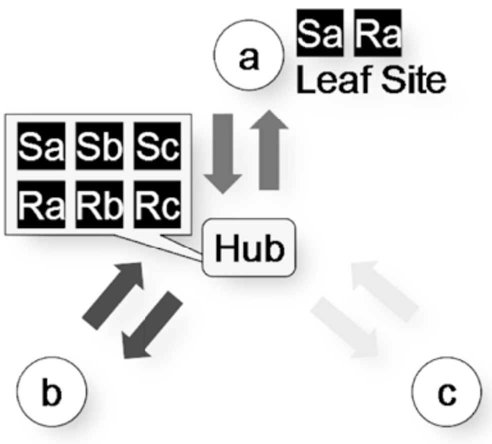


Figure 1 Star configuration satellite network

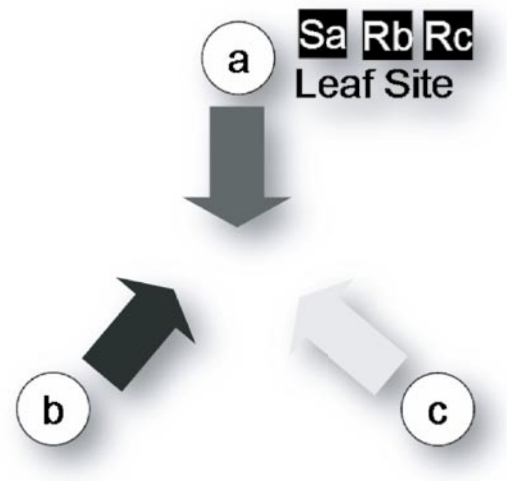


Figure 2 UDL mesh satellite network

This paper proposes an approach to further reduce the overhead of headers in a DVB-S link using ROHC. The performance characteristics of ROHC are evaluated over real satellite links. The second section of this paper discusses the related works on header compression schemes proposed by other researchers. Then it is followed by an overview of the software implementation of ROHC compressor and decompressor. The fourth section of this paper discusses software parameters and hardware configuration used in the experiment. Subsequently, results of the experiment and conclusions are presented in the last two sections.

2 Related Works

Over the past 15 years, there are many proposed header compression schemes. The original Van Jacobson compression scheme (VJHC) [4] was developed to reduce the TCP/IP packet overhead to an average of 4-5 bytes per packet. It does not support compression of IP/UDP headers because at that time UDP traffic was very low. This scheme employs delta compression, where the value of changing fields will be sent, helping to minimize the number of bits sent. This scheme relies on the TCP recovery mechanism to recover from errors in the compressor context due to bit errors and residual error arising from packet losses on the link. This scheme is only suitable for less error-prone links such as terrestrial links. It is because one corrupted delta header can cause all subsequent packets to be misinterpreted.

Subsequently, a general IP header compression scheme named IP header compression (IPHC) [5] was proposed to improve on VJHC. Unlike VJHC, IPHC supports compression of UDP packets. This header compression scheme essentially uses a similar mechanism of delta compression as found in VJHC. The difference between these two header compression schemes is that IPHC uses its own feedback mechanism to recover from error conditions rather than depending on the TCP recovery mechanism. IPHC is suitable for links with strong link error checksums but it is not robust enough to support links with high bit error rates, high losses, and long round trip times.

Compressed Real Time Protocol (CRTP) [6] is a header compression scheme for IPv4/UDP/RTP headers. CRTP can compress 40-byte IPv4/UDP/RTP headers to a minimum of 4 bytes when UDP checksum is enabled. If the UDP checksum was not enabled, CRTP could compress the header to a minimum of 2 bytes. CRTP used explicit signaling messages from the decompressor to the compressor for error recovery. Similar to IPHC, CRTP also did not perform well on lossy links with long round trip time.

To address the high BER and long RTTs on satellite links, an efficient and robust compression scheme is needed. The header compression schemes mentioned above were not able to satisfy this criteria. RObust Header Compression (ROHC)[7] was developed and standardized by the ROHC Working Group within IETF to address these issues. RObust Header Compression (ROHC) is a framework to compress headers of IP packets over error prone links using suitable feedback mechanism. It is a flexible framework that can compress various types of headers. Each ROHC profile defines the required methods to handle the compression and decompression of a specific type of header chains. A stream of related packets is tracked using a context. When encountering a fresh stream of packets, a ROHC compressor creates and initializes a new context with static and dynamic context information using the information derived from the header fields of these packets. At the initial stage, packets belonging to the same context are sent with the least efficient header format. When the compressor is confident that the decompressor has enough information to construct full header with only dynamic fields of header, the compressor will switch to a higher state sending only dynamic fields to update a context.

The ROHC framework defines 3 operational mode, namely unidirectional, bidirectional optimistic and bidirectional reliable. Unidirectional mode is best used when no return path is available to send feedback. In bidirectional optimistic mode, a ROHC compressor may switch to higher state of compression when it is confident that the decompressor has sufficient information to decompress a packet using only dynamic fields or when it receives a positive feedback from the ROHC decompressor. This mode has the potential of realizing the most efficient bandwidth utilization especially when the round trip time of the link is high. When ROHC compressor is operating in bidirectional reliable mode, the compressor can only switch to a higher state when it is informed via a positive feedback message from the decompressor. Each context updating packet is protected by CRC in this mode, thus this mode is the most error-resilient among the three modes.

3 ROHC over ULE Stream

3.1 Implementation of the ROHC Framework over ULE Stream

Encapsulation overheads in MPEG2-TS and ULE are significant for IP packets with small payload size. This paper presents an approach to reduce the overhead of carrying IP packets by applying RObust Header Compression (ROHC) [7] over the ULE stream.

Implementation of the ROHC framework over ULE stream discussed in this paper is an extension to our earlier work [8]. From the software point of view, the bidirectional ULE encapsulator was extended to include the capability to compress packets using ROHC as well as to decompress ROHC compressed packets by means of an external library librohc. Figure 3 depicts the software architecture of the system. Decapsulation of MPEG2-TS frames and

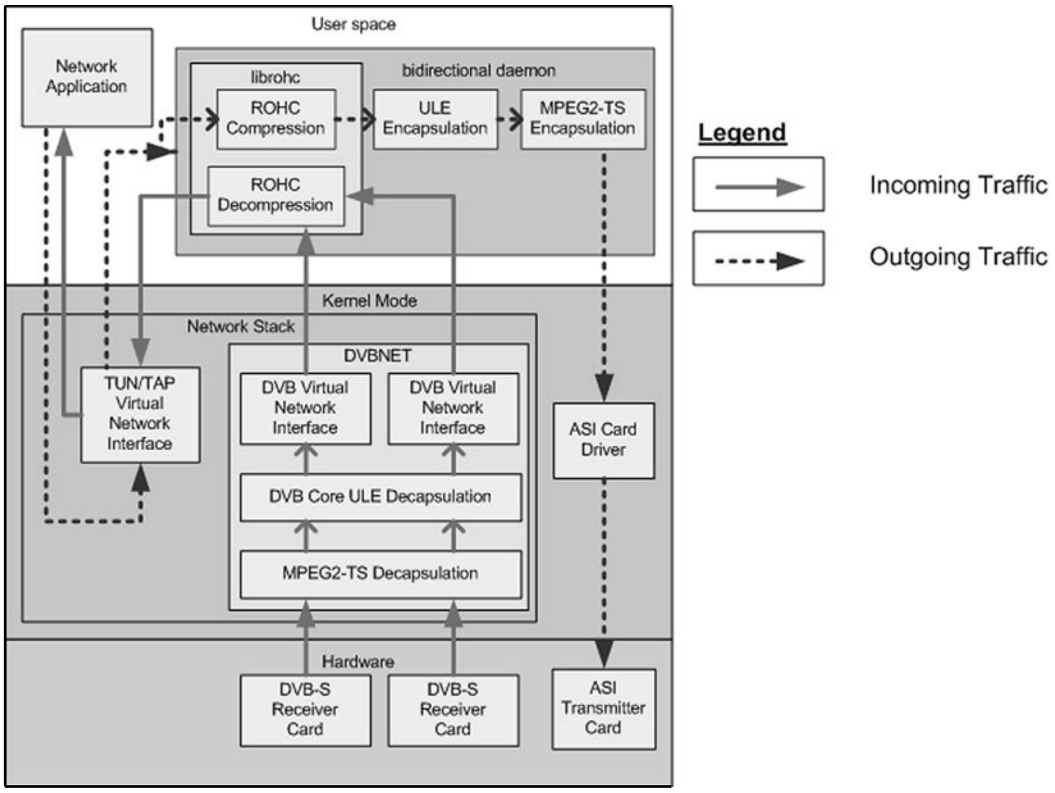


Figure 3 Interaction of bidirectional with other components of the system

ULE SubNetwork Data Unit (SNDU) are done by the linux kernel dvb_net subsystem. Each logical channel of the MPEG2-TS stream is indicated by PID in the MPEG2-TS frame header and represented as a DVB virtual network interface. The bidirectional daemon performs decompression on packets received from these virtual network interfaces before sending them to the tun/tap virtual network interface. Figure 3 shows a simplified version of the overall process and does not take uncompressed streams into consideration.

From then on, the network stack of Linux kernel performs the necessary processing and will deliver the packet to network applications as appropriate. For the transmission of IP packets, the compression of the packets and encapsulation of packets using ULE header and MPEG2-TS frames are done within the bidirectional daemon. These MPEG2-TS frames are then sent to ASI transmitter card. The next subsections will explain the process involved in the compression of IP header and decompression of ROHC compressed packets.

3.1.1 Implementation of ROHC Compressor Framework

The pseudocode of ROHC compressor framework is depicted in Figure 4. When a compressor receives a packet, it first looks for the most suitable ROHC profile to compress the packet. The only criterion for the most suitable ROHC profile is the efficiency of compression for the given packet. For example, given a TCP packet and that the compressor supports TCP compression profile and IP compression profile, both profiles can be selected to compress the TCP packet. However, the TCP compression profile is more suitable since it can compress the TCP header as well as the IP header of the packet, whereas the IP compression profile can only compress the IP header but not the TCP header. If no suitable compression profile can be found, the packet will be returned without compression. This is not to be confused with the uncompressed profile specified in RFC 3095[7].

Once the most suitable compression profile is found, a search is conducted among existing contexts with similar profiles to look for a context that the packet belongs to. If no matching context is found, a new context is created using the information derived from this packet with the help of compression profile. The compressor then encodes the Context Identifier (CID) using the information from the context. The compressor then looks for any ROHC feedback that can be piggybacked. ROHC feedback will be inserted

```
WHILE there is outgoing packet THEN
    IF suitable profile is found for the packet THEN
        IF matching context is not found for the packet THEN
            Create a new context
        END IF
        Code CID using the context
        IF ROHC feedback is found THEN
               Piggyback ROHC feedback
           END IF
           Compress packet header using the context and profile
           Concatenate compressed header and payload
    END IF
    Send outgoing packet
END IF
```

Figure 4 Pseudocode of ROHC compressor framework

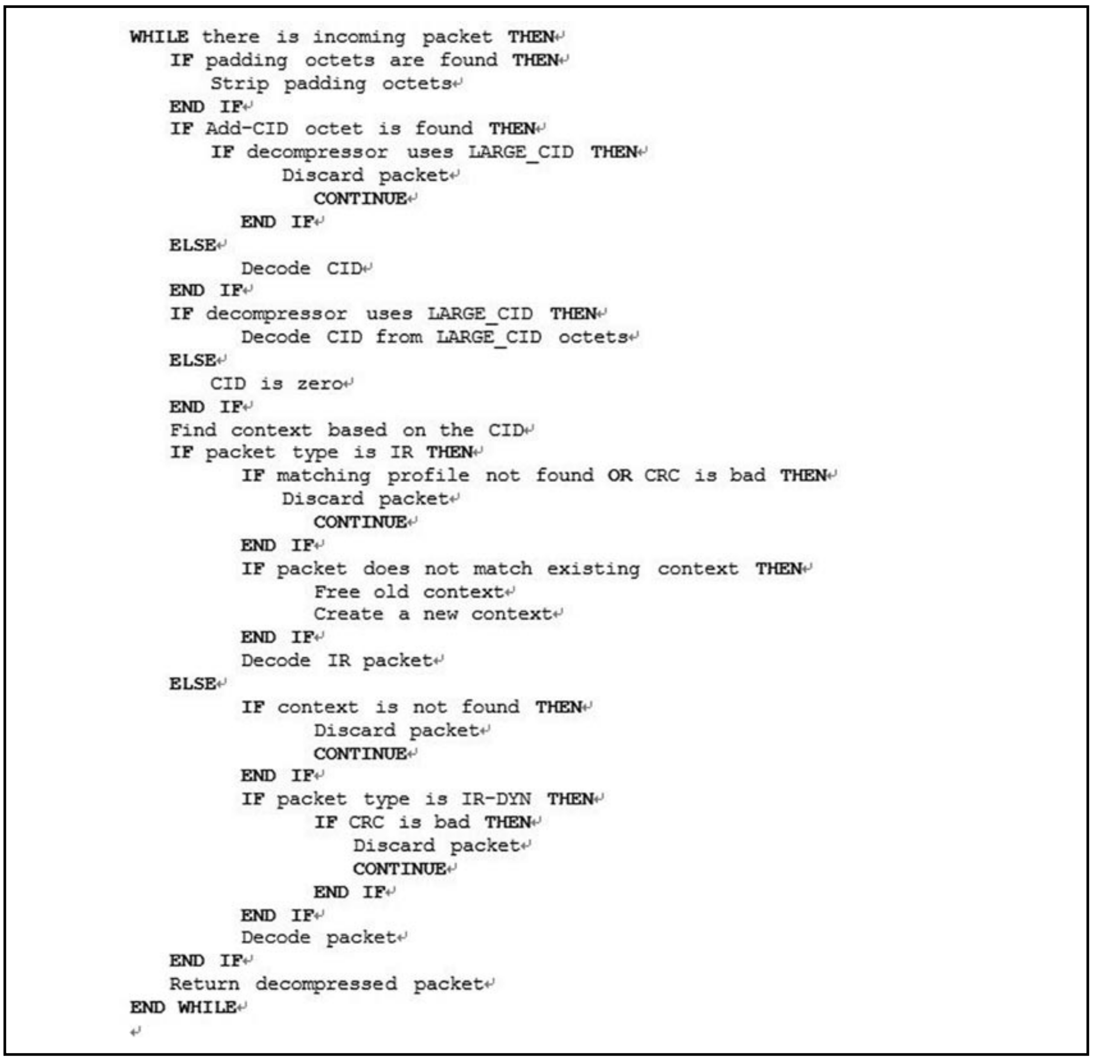
into packet if found.

Armed with the right context and compression profile, the compressor will perform profile specific compression on the packet. In the end, the portion of packet that cannot be compressed will be concatenated to the compressed header to form a compressed packet.

3.1.2 Implementation of ROHC Decompressor Framework

The ROHC decompressor framework is more complicated than the ROHC compressor framework. Its pseudocode is depicted in Figure 5. When a decompressor receives a compressed packet, it strips all padding bytes if present. If it encounters an Add-CID octet, then it needs to decode CID of the packet. If the Add-CID octet is encountered when the ROHC channel uses LARGE-CID, the packet is considered faulty and will be discarded. Then decompressor may encounter ROHC feedback. ROHC feedback can only be decoded if the decompressor is associated with a compressor. Otherwise the whole packet will have to be discarded because the decompressor cannot determine the length of the ROHC feedback.

At this stage, the decompressor will encounter LARGE-CID octets if it is using LARGE-CID and will



```
WHILE there is incoming packet THEN
    IF padding octets are found THEN
        Strip padding octets
    END IF
    IF Add-CID octet is found THEN
        IF decompressor uses LARGE_CID THEN
               Discard packet
                  CONTINUE
           END IF
    ELSE
           Decode CID
    END IF
    IF decompressor uses LARGE_CID THEN
           Decode CID from LARGE_CID octets
    ELSE
        CID is zero
    END IF
    Find context based on the CID
    IF packet type is IR THEN
           IF matching profile not found OR CRC is bad THEN
               Discard packet
                  CONTINUE
           END IF
           IF packet does not match existing context THEN
                  Free old context
                  Create a new context
           END IF
           Decode IR packet
    ELSE
           IF context is not found THEN
                  Discard packet
                  CONTINUE
           END IF
           IF packet type is IR-DYN THEN
                  IF CRC is bad THEN
                      Discard packet
                      CONTINUE
                  END IF
           END IF
           Decode packet
    END IF
    Return decompressed packet
END WHILE
```

Figure 5 Pseudocode of ROHC decompressor framework

have to decode the CID of the packet. Or else the decompressor will have to assume CID 0 is being used if SMALL-CID is used and no Add-CID octet was found previously. A search for existing context will be conducted based on the decoded CID. The packet type is determined at this stage. If the packet type is Initialization and Refresh (IR), the profile that is used to decode this packet is searched for. If the decompressor does not support the profile, then the packet will be discarded. Otherwise a CRC check will be performed and if the CRC check passes, the compressed packet will be compared against existing context for the CID, if there is one, to determine if it matches existing context. If it does not match the existing context, the compressor must have reused the CID for a new context, thus the old context will have to be deleted and a new context based on current IR packet will be created. The decompressor then performs profile specific decompression of the IR packet.

If, however, the packet is not an IR packet, then the packet will be discarded if the earlier effort to look for existing context failed. If the packet type is Initialization and Refresh-Dynamic (IR-DYN), CRC check will be conducted. Decoding of the header will then be performed based on the context and profile of the packet. The decompressed header and payload of the compressed packet will be concatenated to form the uncompressed packet.

4 Setup of Experiment

4.1 Physical Environment

The experiment was conducted through C-Band satellite link provided by JSAT's JCSAT-3A transponder. The participant sites were Shonan Fujisawa Campus (SFC) at Keio University and Universiti Sains Malaysia (USM) in Penang. The satellite link from USM to SFC had a link capacity of 3.507Mbps, while the satellite link from SFC to USM used a shared link over an ASI multiplexer as depicted in Figure 6. The UDL mesh bridge machine at SFC was configured to transmit data at 3Mbps over the 12Mbps link.

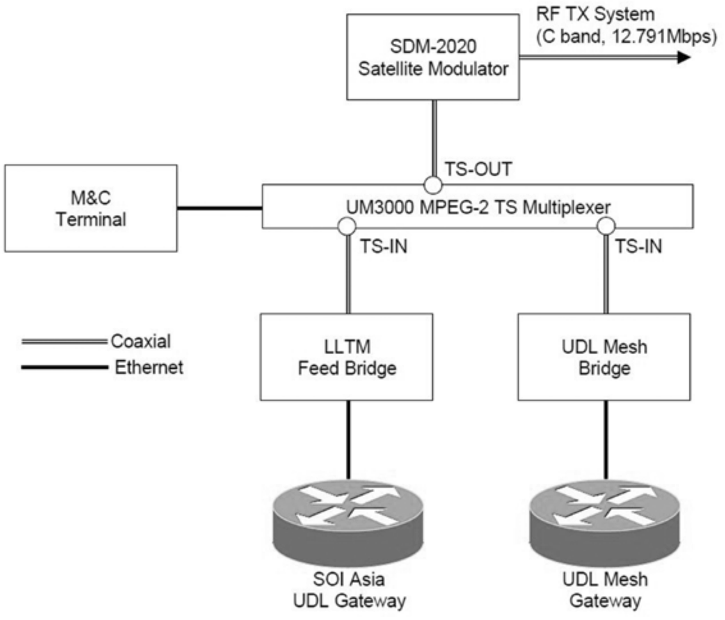


Figure 6 Configuration of UDL mesh gateway at SFC campus

4.2 Parameters of ULE Encapsulator

There are two mechanisms to encapsulate ULE SubNetwork Data Unit (SNDU) within MPEG2-TS frames known as padding and packing. In padding mode, after the ULE SNDU is encapsulated within a MPEG2-TS frame, the unused portion of the MPEG2-TS frame is padded with padding bytes and sent immediately. Whereas when packing mode is used, the unused portion of MPEG2-TS frame will be filled with the subsequent ULE SNDU if a ULE SNDU is received before packing threshold expires. Based on the results from previous experiments, setting packing threshold of 1 millisecond packing threshold was not ideal for throughput in certain cases. While using 5, 10 and 20 milliseconds packing threshold were ideal for throughput, higher latencies were incurred on these packing thresholds, by default, the experiment was conducted using packing mode with a 3 milliseconds packing threshold as it was the best tradeoff in terms of throughput and latency. Padding mode was not used as the default setup because of its inefficiency. The results in Figure 7 were obtained by running the experiment using padding mode with various packet sizes. Even though the compressed stream achieved better results when compared to uncompressed stream in padding mode, it still yielded worse results than an uncompressed stream in packing mode whenever compressed packet sizes slightly exceed a multiple of 184 bytes (one MPEG2-TS frame).

4.3 Parameters of the ROHC Compressor

The ROHC compressor used in this experiment only supports UDP profile operating in unidirectional

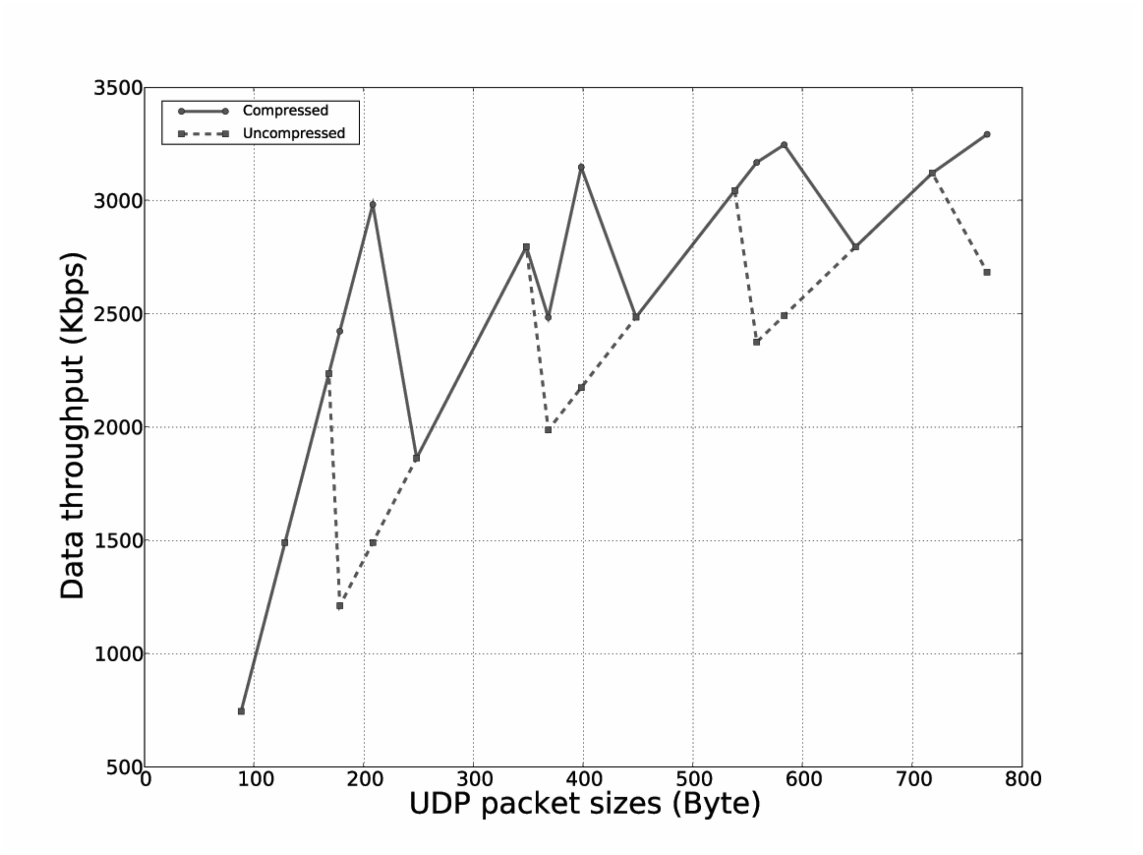


Figure 7 Data throughput of compressed and uncompressed streams in padding mode

mode. The compressor was configured to send 3 **Initialization and Refresh (IR)** packets before moving to a higher state. The compressor was set to perform periodical down-transition to the IR state after transmitting 50 **Second Order (SO)** packets consecutively to ensure that the context is synchronized properly. To achieve highest efficiency, the ROHC channel was configured to use SMALL_CID to reduce the overhead of **Context Identifier (CID)**. A CID of zero is sent implicitly and never shows up as part of header.

When a ROHC compressor is operating at the highest efficiency, it is capable of reducing the headers of IP packet down to 1 byte. However, for IPv6 UDP datagrams, the UDP packet must be coupled with a 2 octets UDP checksum as required by IPv6. Thus, the smallest compressed header for IPv6/UDP stream will always occupy 3 bytes. Figure 8 shows the format of the most compact compressed header.

0	1	2	3	4	5	6	7
0	SN				CRC		
UDP Checksum(2 octets)							

Figure 8 Packet format of UDP/IPv6 UO-0 header

When the ROHC compressor is in the IR state, it has to initialize the context with static and dynamic information using the IR packet. The IR packet is slightly bigger than the original uncompressed packet. Figure 9 shows the structure of the IR packet for a UDP/IPv6 stream. The notation of UDP/IPv6 follows the style proposed in the RFC for ROHC [7] and it denotes UDP over IPv6. This type of notation is applied throughout this paper. Generic extension header list may span more than 1 byte depending on the UDP/IPv6 header. The IR packet is at least 50 bytes.

0	1	2	3	4	5	6	7
1	1	1	1	1	1	0	D
Profile = 0 × 2 (1 octet)							
CRC (1 octet)							
Version = 6							
Flow Label (20 bits)							
Next Header (1 octet)							
Source Address (128 bits)							
Destination Address (128 bits)							
Source Port (2 octets)							
Destination Port (2 octets)							
Traffic Class (1octet)							
Hop Limit (1 octet)							
Generic extension Header list (1 octet)							
UDP Checksum (2 octets)							
SN (2 octets)							

Figure 9 Packet format of UDP/IPv6 IR header

By making several assumptions, a theoretical model of achievable data throughput for ROHC compressed UDP/IPv6 stream and uncompressed UDP/IPv6 stream was developed. Let us denote the bandwidth of the link in bps as ***Bandwidth***, thus the number of MPEG2-TS frames transmitted per second, denoted by ***TS***, can be calculated using the following formula:

$$\boldsymbol{TS} = \frac{\boldsymbol{Bandwidth}}{188 \times 8} \tag{1}$$

Each MPEG2-TS frame is 188 bytes and out of these 188 bytes, 184 bytes were used to carry its payload, the ULE SNDU. Assuming that the overhead of the Payload Pointer (PP) of MPEG2-TS frame is negligible, the number of bytes that can be used by an ULE SNDU, denoted by ***ULE***, can be deduced from the following formula:

$$\boldsymbol{ULE} = \boldsymbol{TS} \times 184 \tag{2}$$

It should be noted that the overhead of PP is not negligible when the size of payload gets very small. Each ULE packet adds 8 bytes of overhead with four bytes taken up by the ULE header and another 4 bytes used for 32-bit CRC. For a typical UDP/IPv6 datagram, 40 bytes are taken up by IPv6 header and 8 bytes are used for UDP header. Let us denote the payload of UDP packet in bytes as ***Payload***, hence the number of uncompressed UDP/IPv6 packets that can be sent in 1 second, denoted by $\boldsymbol{uncompressed}_{countIPv6}$, and achievable data throughput in bps, denoted by $\boldsymbol{uncompressed}_{bpsIPv6}$, can be derived as follows:

$$\boldsymbol{uncompressed}_{countIPv6} = \frac{\boldsymbol{ULE}}{(\boldsymbol{Payload} + 48 + 8)} \tag{3}$$

$$\boldsymbol{uncompressed}_{bpsIPv6} = \boldsymbol{uncompressed}_{countIPv6} \times \boldsymbol{Payload} \times 8 \tag{4}$$

Assuming that the compressor directly transits from the Initialization and Refresh (IR) state to the Second Order (SO) state without going through First Order (FO) state, the average size of ROHC compressed headers, $\boldsymbol{average}_{ROHC}$, can be calculated using the following formula:

$$\boldsymbol{average}_{ROHC} = \frac{\boldsymbol{IR}_{count} \times \boldsymbol{IR}_{size} + \boldsymbol{SO}_{count} \times \boldsymbol{SO}_{size}}{\boldsymbol{IR}_{count} + \boldsymbol{SO}_{count}} \tag{5}$$

where:

$\boldsymbol{IR}_{count}$ is the number of IR packets sent by the compressor before moving to the SO state.

$\boldsymbol{SO}_{count}$ is the number of SO packets sent by the compressor before down-transiting to the IR state.

$\boldsymbol{IR}_{size}$ is the size of the IR header in bytes. It is assumed to be 50 bytes for a typical UDP/IPv6 stream as depicted in Figure 9.

$\boldsymbol{SO}_{size}$ is the size of the SO header in bytes. It is assumed to be 3 bytes for a typical UDP/IPv6 stream as depicted in Figure 8.

The number of compressed UDP/IPv6 packets that can be sent in 1 second, denoted by $\boldsymbol{compressed}_{countIPv6}$, and achievable data throughput in bps, denoted by $\boldsymbol{compressed}_{bps}$, can be derived using the following formulae:

$$\boldsymbol{compressed}_{countIPv6} = \frac{\boldsymbol{ULE}}{(\boldsymbol{Payload} + \boldsymbol{average}_{ROHC} + 8)} \tag{6}$$

$$\boldsymbol{compressed}_{bpsIPv6} = \boldsymbol{compressed}_{countIPv6} \times \boldsymbol{Payload} \times 8 \tag{7}$$

Using the assumptions stated above, Figure 10 shows a comparison of maximum theoretical data throughput between compressed and uncompressed UDP/IPv6 streams over ULE/MPEG2-TS using packing mode over the 3.507Mbps DVB-S link. Figure 11 shows the percentage of data throughput gain for a ROHC compressed stream over an uncompressed stream.

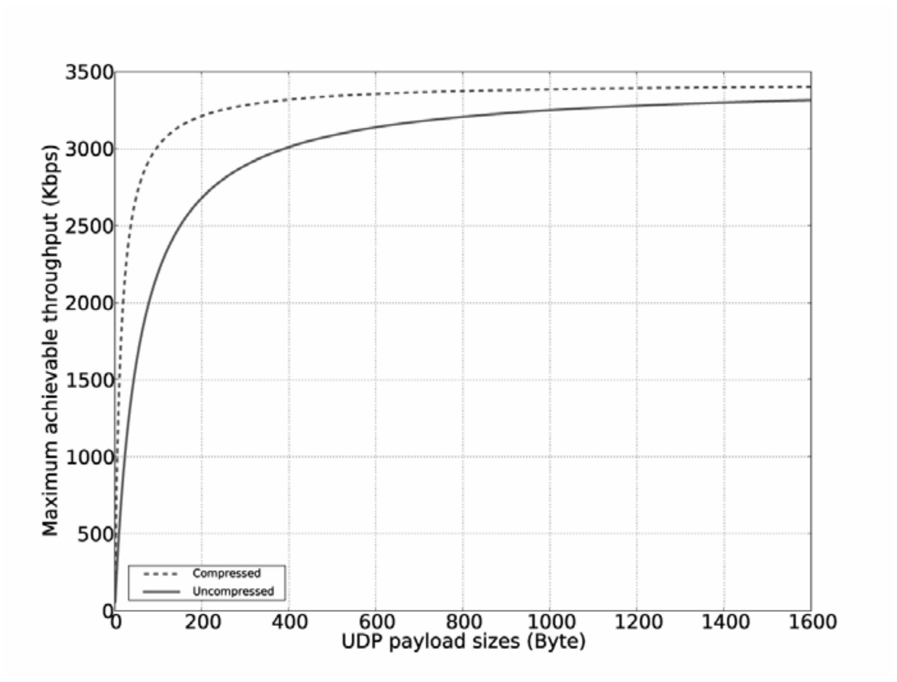


Figure 10 Comparison of maximum theoretical data throughput between uncompressed and compressed UDP/IPv6 streams

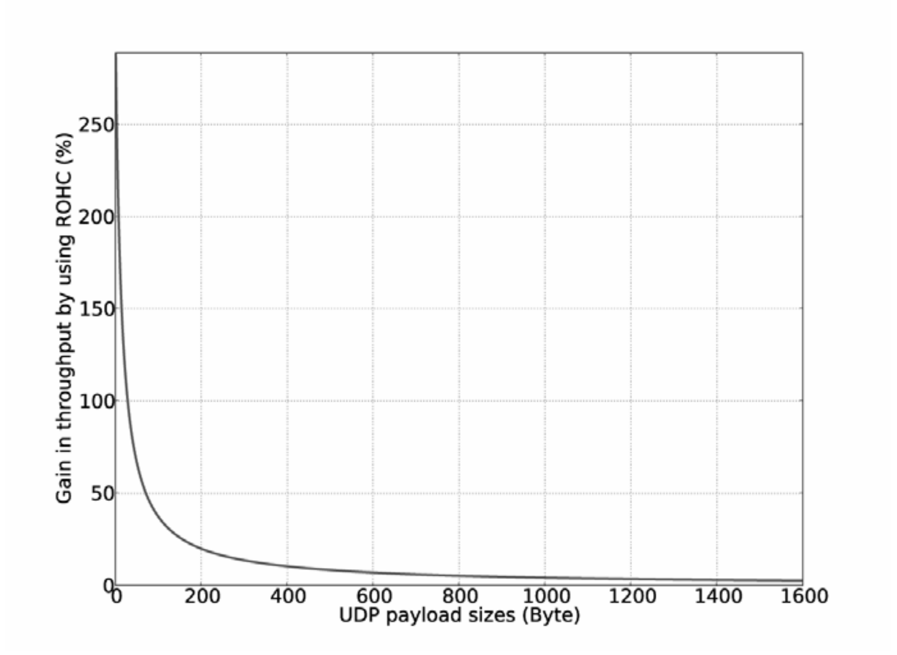


Figure 11 Data throughput gain of ROHC compressed UDP/IPv6 stream

4.4 Experiment Details

The results of the experiment were gathered using iperf, a client-server software for throughput measurement. The iperf client ran in the gateway located at USM, while the iperf server ran in the gateway located at SFC. The ULE encapsulator operated in packing mode. Tests were conducted with UDP payload sizes of 40 bytes, 80 bytes, 120 bytes, 160 bytes, 200 bytes, 300 bytes, 512 bytes and 1024 bytes. More samples were taken for smaller payload sizes since significant changes occur in this region of the graph as depicted in Figure 10 and 11.

5 Results

For Figure 12 - 14, tests were conducted incrementally by instructing the iperf client to send data stream at 100kbps up to 3500kbps with an increment of 100kbps for consecutive tests to determine the point where data throughput peaked for both compressed and uncompressed stream. The data stream generation rate and the measured data throughput were normalized against the bandwidth of the link. Packet loss rates (PLR) became non-zero when measured maximum data throughput was obtained. Increasing packet generation rates beyond that saturation point will lead to packet losses.

For payload sizes of 40 bytes, 80 bytes and 120 bytes, compressed data throughput saturated at 828 kbps (23.7%), 1656 kbps (47.3%) and 2315 kbps (66.1%) respectively. For these payload sizes, uncompressed data throughput saturated at 746 kbps (21.3%), 1547 kbps (44.2%) and 2238 kbps (63.9%) respectively. The benefit of ROHC for these packet sizes was minimal especially for payload size of 120 bytes where compressed traffic yielded only 2.2 % increase in data throughput. The measured data throughput was significantly lower compared to the available bandwidth of the link when the UDP payload sizes were small. This was caused by the overheads introduced by MPEG2-TS frame, ULE SNDU, IPv6, and UDP headers, which took up a significant portion of the available link bandwidth compared to the actual data payload size.

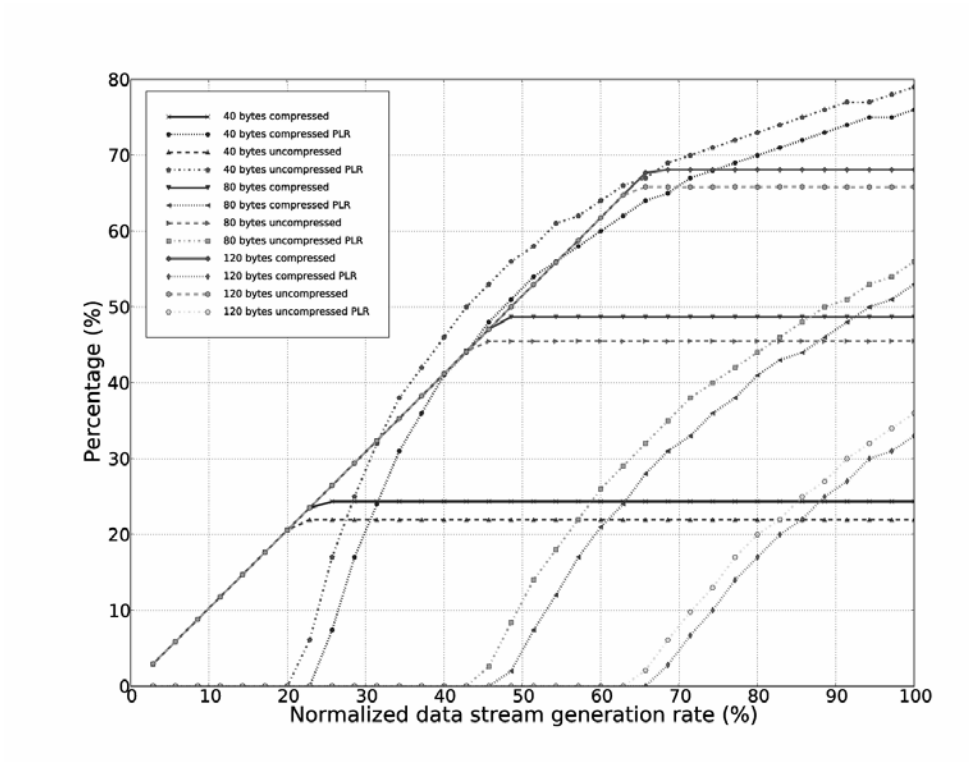


Figure 12 Normalized data throughput of compressed and uncompressed UDP/IPv6 streams with 40, 80 and 120 bytes of UDP payload

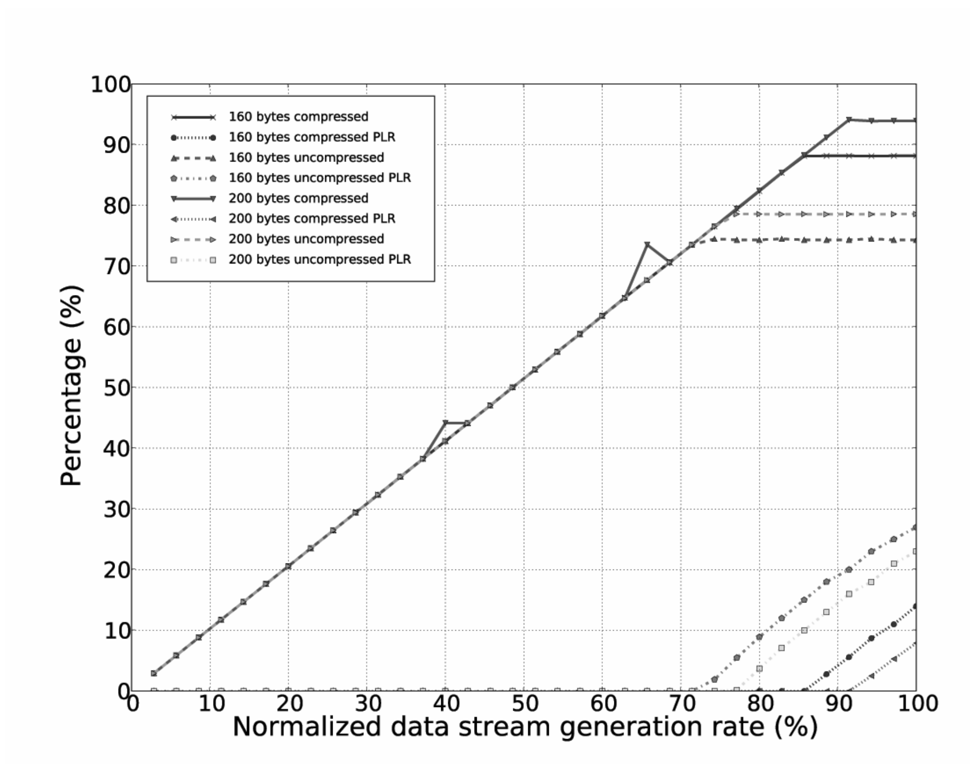


Figure 13 Normalized data throughput of compressed and uncompressed UDP/IPv6 streams with 160 and 200 bytes of UDP payload

Results for payload size of 160 bytes were marginally better. The highest data throughput recorded for uncompressed traffic approximated the result from the theoretical model as shown in Figure 15. The highest data throughput recorded for compressed traffic was about 19% better than highest data throughput of uncompressed traffic. Likewise, packet loss rates were marginally lower. Unlike the results in Figure 12, compressed data throughput with 160 bytes payload was better than uncompressed data throughput with 200 bytes payload.

For payload sizes of 300 bytes, 512 bytes and 1024 bytes, the highest data throughput for compressed traffic was greater than 90 % of link capacity. Percentage of data throughput gains decreased gradually as the payload size increased. However, for a payload size of 1024 bytes, data throughput of uncompressed traffic was less than that of 512 bytes. The reason for this is unknown although several runs of the experiment gave consistent results.

Figure 15 is a summary of Figure 12 - 14. It shows that the measured data throughput of compressed traffic and uncompressed traffic from DVB-S link were comparable to the results from theoretical model for UDP payload of 200 bytes or greater.

Contrary to the results from the theoretical model, the measured results from DVB-S link did not exhibit higher gains when payload sizes were small (< 200 bytes), possibly due to interactions between the hardware device and software drivers. Interestingly, Figure 16 shows that the gain achieved for UDP payload of 1024 bytes was higher than that of theoretical model. The measured data throughput of uncompressed traffic for that payload size was slightly lower than expected, while the measured

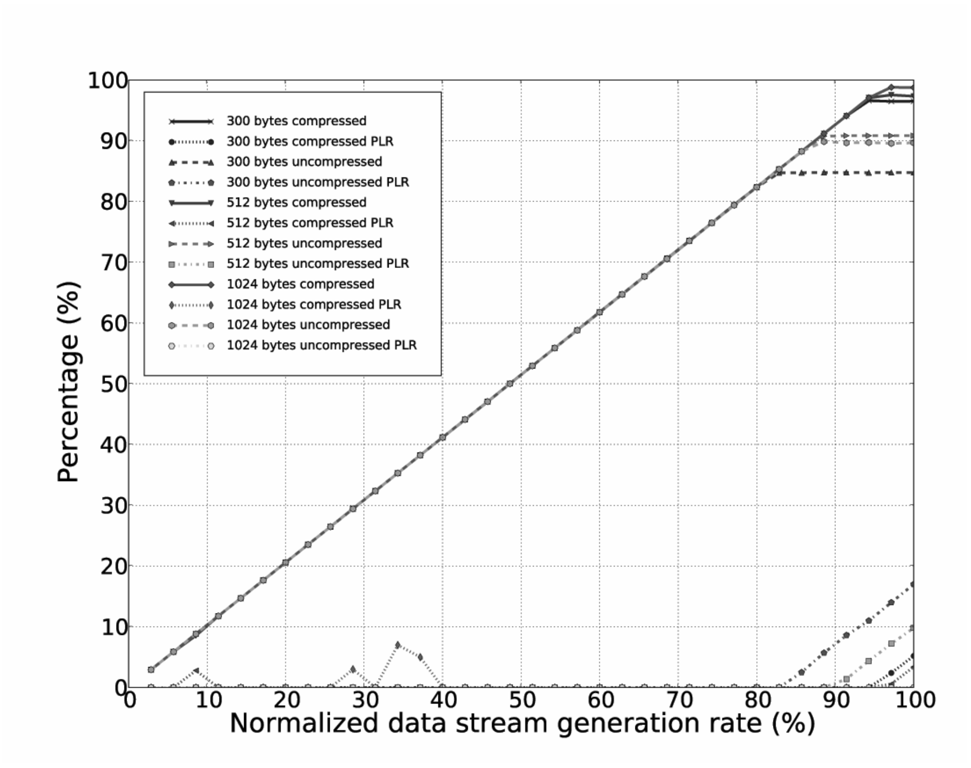


Figure 14 Normalized data throughput of compressed and uncompressed UDP/IPv6 streams with 300, 512 and 1024 bytes of UDP payload

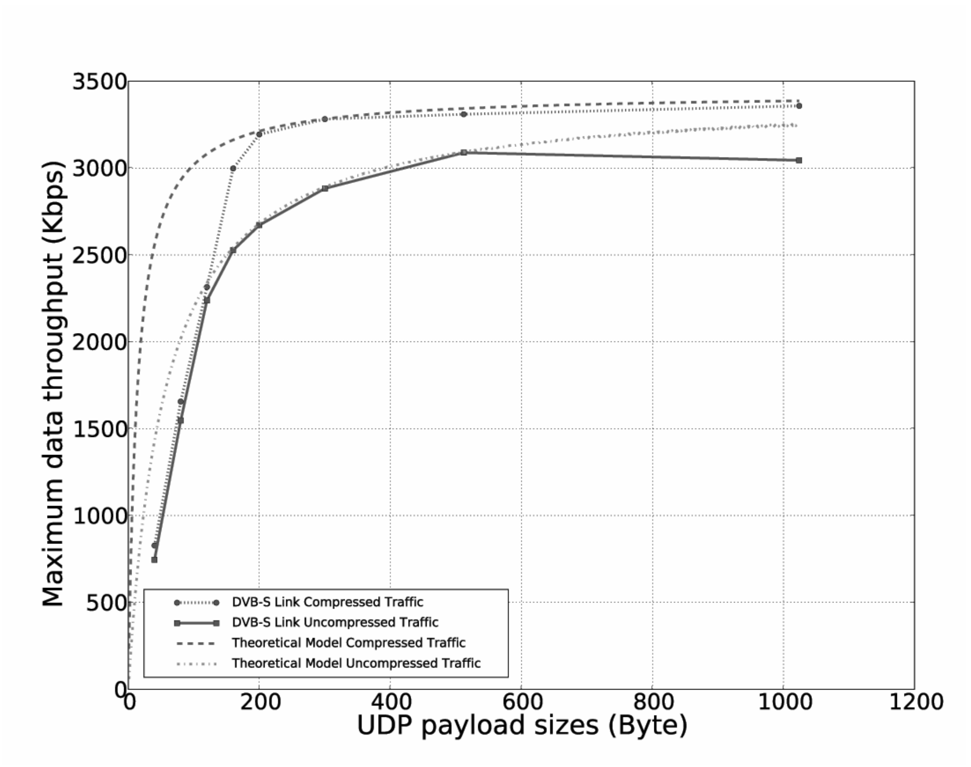


Figure 15 Comparison of maximum data throughput between theoretical model and DVB-S link

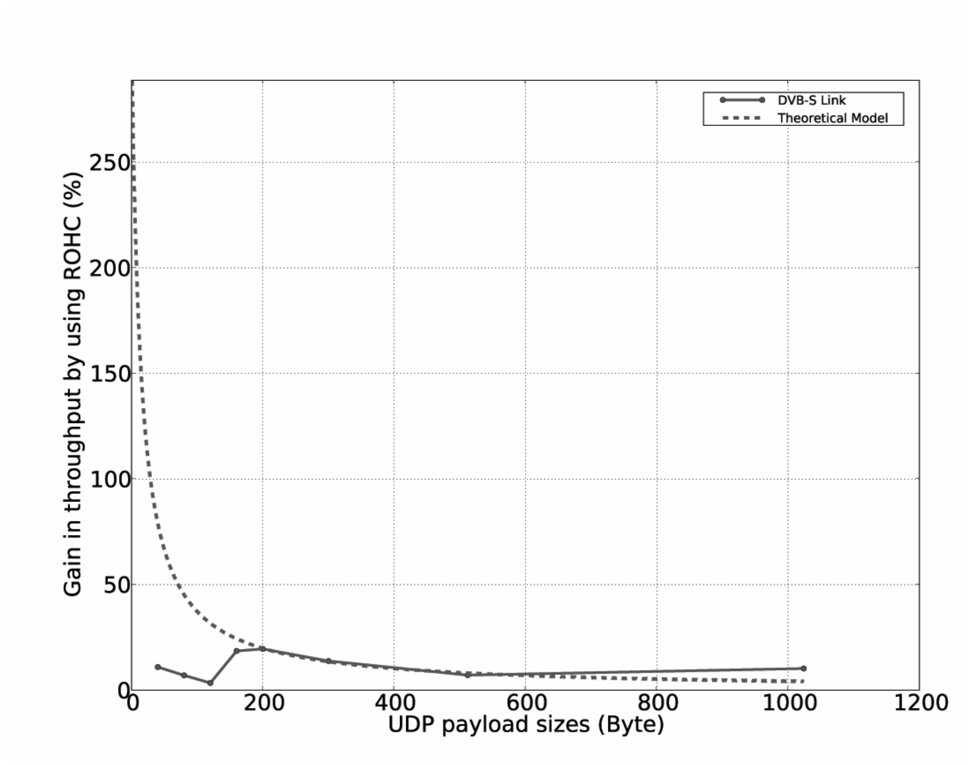


Figure 16 Comparison of data throughput gains between theoretical model and DVB-S link

data throughput for compressed traffic was close to the expected result from the theoretical model. This combination led to a higher than expected data throughput gain for the UDP payload size of 1024 bytes.

6 Conclusion

The implementation of UDL mesh network, ULE and ROHC were intended to improve the utilization of satellite bandwidths. However, the implementation of ULE encapsulator and ROHC framework deviated from the theoretical model when payload sizes were small (< 200 bytes). Due to intricacies of interaction between hardware and software, additional fine tuning must be done to achieve results closer to the theoretical model when payload sizes are small. Modifications to the theoretical model to account for overhead introduced by the Payload Pointer should also improve the correlation between the theoretical and experimental results. Nonetheless, the experiments show that the ROHC over ULE performance is very close to the predicted behavior for UDP payloads of 200 bytes or greater. An earlier study to predict the performance of ROHC over ULE shows that efficient bandwidth utilization can be achieved for various types of RTP traffic [9]. Future work include the implementation and testing of more advanced profiles (e.g., RTP, TCP) which is expected to improve the achievable data throughput of ROHC over ULE even more for such real time audio and video streams.

7 Acknowledgment

The authors wish to thank the AI3 Project for the satellite link bandwidth used in the experiments. Implementation of the ROHC framework by Lulea

University of Technology was used as reference for the implementation of librohc used in the experiments. This research was partially funded by Ministry of Science, Technology and Innovation (MOSTI) Malaysia via e-Science grant titled Improving Performance of Unidirectional Link (UDL) Mesh Network via Header Compression (Project 01-01-05-SF0036).

References

[1] T.C. Wan, “Interactive wide area collaboration via unidirectional links,” *Proceedings APAN Conference*, Beijing, P.R. China, 2000.

[2] G.Fairhurst and B.Collini-Nocker, “Unidirectional Lightweight Encapsulation (ULE) for Transmission of IP Datagrams over an Mpeg-2 Transport Stream (TS),” *RFC 4326*, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4326.txt

[3] C.H. Teh, T.C. Wan, and R.Budiarto, “A comparison of IP datagrams transmission using MPE and ULE over MPEG-2/DVB networks,” In *Proceedings Fifth Int’l Conference on Information, Communications, and Signal Processing (ICICS 2005*), pp. 1173-1177. Bangkok, Thailand, 2005.

[4] 4V.Jacobson, “Compressing TCP/IP headers for low-speed serial links,” *RFC 1144*, 1990. [Online]. Available: http://tools.ietf.org/html/rfc1144

[5] M.Degermark, B.Nordgren, and S.Pink, “IP Header Compression,” *RFC 2507*, 1999. [Online]. Available: http://tools.ietf.org/html/rfc2507

[6] T.Koren, S.Casner, J.Geevarghese, B.Thompson, and P.Ruddy, “Enhanced Compressed RTP (CRTP) for links with high delay, packet loss and reordering”, *RFC 3545*, 2003

[7] C.Borman, C.Burmeister, M.Degermark, H.Fukushima, H.Hannu, L.-E. Jonsson, R.Hakenberg, T.Koren, K.Le, Z.Liu, A.Martensson, A.Miyazaki, K.Svanbro, T.Wiebke, T.Yoshimura, and H.Zheng, “RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed”, *RFC 3095*, 2001

[8] W.C. Ang, N.A. binti Yusri, C.W. Tan, and T.C. Wan, “Implementation of IP mesh networks using ULE protocol over DVB-S links,” *Proceedings Int’l Conference on Computing and Informatics, ICOCI 2006*, Kuala Lumpur, Malaysia, Jun. 2006.

[9] M.Sooriyabandara, M.Forrest, and G.Fairhurst, *Broadband Satellite Comunication Systems and the Challenges of Mobility*, IFIP International Federation for Information Processing. Springer Boston, 2005, Vol.169, pp.81-90, 2005. [Online]. Available: http://www.springerlink.com/content/e5pheghwjeuburqa/